Created by Ahsan Arif

IN Function

The IN function helps reduce the need to use multiple *OR* conditions.

The syntax for the IN function is:

```
SELECT columns
FROM tables
WHERE column1 in (value1, value2, .... value_n);
```

This SQL statement will return the records where column1 is value1, value2..., or value_n. The IN function can be used in any valid SQL statement - select, insert, update, or delete.

Example #1

The following is an SQL statement that uses the IN function:

```
SELECT *
FROM suppliers
WHERE supplier_name in ( 'IBM', 'Hewlett Packard', 'Microsoft');
```

This would return all rows where the supplier_name is either IBM, Hewlett Packard, or Microsoft. Because the * is used in the select, all fields from the suppliers table would appear in the result set.

It is equivalent to the following statement:

```
SELECT *
FROM suppliers
WHERE supplier_name = 'IBM'
OR supplier_name = 'Hewlett Packard'
OR supplier_name = 'Microsoft';
```

As you can see, using the IN function makes the statement easier to read and more efficient.

Example #2

You can also use the IN function with numeric values.

```
SELECT *
FROM orders
WHERE order_id in (10000, 10001, 10003, 10005);
```

This SQL statement would return all orders where the order_id is either 10000, 10001, 10003, or 10005.

It is equivalent to the following statement:

```
SELECT *
FROM orders
WHERE order_id = 10000
OR order_id = 10001
OR order_id = 10003
OR order_id = 10005;
```

Example #3 using "NOT IN"

The IN function can also be combined with the NOT operator.

For example,

```
SELECT *
FROM suppliers
WHERE supplier_name not in ( 'IBM', 'Hewlett Packard', 'Microsoft');
```

This would return all rows where the supplier_name is **neither** IBM, Hewlett Packard, or Microsoft. Sometimes, it is more efficient to list the values that you do **not** want, as opposed to the values that you do want.